Spike: Task 14

Title: Agent Markmanship

Author: Sam Huffer, 101633177

# Goals / Deliverables

* Basic deliverables:
  + An agent targeting simulation with:
    - An attacking agent (stationary or otherwise)
    - A moving target agent (can move between two points) that shows when it has been hit
    - Several weapons that can successfully hit the target, including:
      * Rifle (fast, accurate)
      * Rocket (slow, accurate)
      * Hand gun (fast, inaccurate)
      * Hand grenade (slow, inaccurate)
* Extensions:
  + Account for rate of fire and effect range (specifically, make a shotgun with a low rate of fire and short but deadly effective range; move the attacker closer before shooting).
  + Splash damage (projectile that explodes on the ground; attacker should take this into account when aiming).
  + A target that can avoid slow projectiles and or move to hiding spots when attacked.

# Technologies, Tools, and Resources Used

* SublimeText (for editing, executing and testing the code)
* Learning materials on Canvas (for instructions and sample code)

# Tasks Undertaken

* I copied the Tactical Steering project from task 11 into the task 14 folder.
* Cleaned up some code, started putting together a skeleton for the methods.
* Added the walls from task 11, tweaking existing methods as needed to accommodate them.
* Added input controls for toggling the obstacles and walls on and off, tweaking existing methods as needed to accommodate that
* Set up target and shooter’s starting positions.
* Set up shooting from those stationary positions, pooling the projectiles rather than instantiating and destroying repeatedly to improve program efficiency. Projectiles die on contact with the target or when they pass outside the bounds of the screen.
* Added inaccuracy within a specified range.
* Added display of being hit, with target going red for 0.1 seconds.
* Set up “moving back and forth” and “wander” movement types for the target
* Set up predictive shooting for the shooter. Tried using a fancy method that uses sine and cosine functions, but that failed miserably. Currently calculates time it’d take the projectile to get to the current / currently-calculated-future position, then finds the target’s new position at that time using d = ut + ½ at2, and wraps position, iterating through this process until it thinks the projectile will be able to get within an acceptable distance of the target.
* Tweaked hiding spots, fleeing, hunting and fov updating to work with a single hunter and evader, and to fit with the this task’s logic for when to move and what movement to do
* Set up explosions for rockets and hand grenades, and proved that I am a massive child.

# Code Snippets

# Instructions for Operating the Code

* I: toggle the display of agents’ force, velocity and net desired change in position.
* N: create a new obstacle in a random but valid position.
* O: toggle obstacles and hiding spots on and off.
* P: pause or un-pause the game.
* R: reposition all obstacles in random but valid positions. Obstacles are automatically repositioned when the window changes size.
* S: Scroll through shooter weapons.
* T: Scroll through target movement types (stationary, moving between two points, evading).
* W: Toggle walls on and off.
* Escape: exit the game.
* Space: Shoot.

# In-Simulation Screenshot

# What I Found Out

* Setting up different projectile types was easy enough. Setting up the movement settings and writing the predictive logic, even with the pursuit method as a basis for the latter, was the tricky bit.
* If target changes its velocity (vector or magnitude) or acceleration (ditto), that can render the predicted position useless, particularly if the projectile isn’t moving fast enough for the inaccuracy to be negligible.